intel®

ARTICLE REPRINT

AR-107

September 1979

Use EPROM 1-chip μCs as Effective 1-shot Lab Aids

121507-001

Use EPROM 1-chip μCs as effective 1-shot lab aids

Single-chip microcomputers with on-board EPROMs can advantageously replace TTL circuitry in laboratory evaluation projects.

Robert H Cushman, Special Features Editor

Suppose you're assigned the task of producing a "quick-and-dirty" tester for a 1-shot application such as evaluating the reliability of a proposed component. Should you use TTL or a 1-chip μC with an on-board EPROM? The EPROM-μC approach now offers many advantages.

Let's be more specific: Your company is developing an automotive system, and your boss wants to know whether a certain component—anything from a solid-state device to an electromechanical relay—will have the required mean time to failure in the system's operating environment. He therefore tells you to whip up a test jig that will cycle the component and count the number of successful operations. Your tester should be reliable enough so that it won't fail during the testing process and compact enough so that it can fit in your lab's small environmental chamber and also attach inconspicuously to automobiles for road tests.

"But don't spend too much time or money on this 1-shot project," your boss admonishes, "not more than a week at most."

Arguments for the 1-chip-μC approach

Confronted with these requirements, you'll find that the use of 1-chip μCs (and also "few-chip" systems; see **box**) for 1-shot lab applications has some irresistible advantages. Based on our recent experience in using an Intel 8748 1-chip μC, we feel that in such applications, a 1-chip device with on-board EPROM is

- The easiest design approach
- The lowest cost approach, in terms of both first cost and follow-on-modification cost
- The fastest solution
- The most reliable solution
- The most compact solution
- The most readily duplicated solution if you require multiple copies
- The solution most likely to prove reusable in future projects.

As we see it, this technique has only two caveats: First, the application must lie within the bandwidth of available 1-chip μCs, which usually means that the signals handled must not range much beyond the audio frequencies. Second, the designer must have a working familiarity with the μC used and have access to and experience with suitable development tools for that device.

Neither of these two restrictions is a hard-and-fast barrier, however. For wider-band signals, you can use outboarded TTL and ECL interfaces. And for the rather small programs involved in these 1-shot applications, you needn't really be an experienced expert in *all* phases of the μC's use; you can often—as we did with the Intel Prompt-48 debugger/programmer—take a simplified approach.

Component life cycler's a good example

The hardware layout of our example project appears in **Fig 1.** Basically, the 8748 provides ON and OFF pulses to drive the device under test (DUT) and then counts the number of successful DUT operations.

As shown, we used this tester on two different types of DUTs: a reed relay and an optointerrupter. For the former, we wanted to find out how many cycles it could survive under varying cycling rates and levels of application vibration. For the latter, we wanted to determine whether the dirt, fumes and grease of various automotive and industrial environments would obscure its optical path.

We chose with care the 8748 features used in the tester, trying to leave free those that we might want for future enhancements. Thus, we drove the DUTs from a pin in the top nibble of Port 2 so the port's bottom nibble would still be available for bus expansion. Likewise, we sensed the DUT's response on one of the 8748's Test inputs, rather than tying up the μC's sole Interrupt input. And we avoided using the data-bus port (Port 0) altogether.

A pushbutton switch connected to the 8748

Is TTL or a 1-chip μC the better choice for a 1-shot lab project?

Reset input starts the test cycling, and a yellow LED indicates when cycling has ceased (presumably because the DUT has failed to respond). (A more elaborate optional indicator—a single 7-segment-plus-decimal-point LED readout—is shown with dashed lines and will be discussed shortly.) The system's as-yet untested power supply is sketched in as a rugged combination of a zener protected by a surge arrestor and backed up by a large capacitor that should keep the tester running through short dips in supply voltage (as can occur in automotive and industrial environments). If we were using the new CMOS 8748 from Intersil (the 87C48), we would also include a trickle-charged backup battery as part of this power supply to allow the 87C48 to retain test results despite power-supply failures.

The software flow diagram for the cycler's basic routines appears in **Fig 2,** where some of the key 8748 instructions used are also called out. All told, about 100 bytes of the 8748's 1k EPROM are required to implement this example.

The tester's initialization routine involves clearing the 8748 data-RAM register string that holds the count resulting from the DUT cycling. If the 8748 I/O pins must be in a certain initial state, implementing that state would also form part of the initialization process.

Test cycling occurs through an endless loop that turns the DUT ON, calls a delay, turns the DUT OFF and calls the delay again. This delay is inserted after both the ON and OFF actions to allow the DUT time to respond; it could be realized through the 8748's built-in timer, but we elected to use a software delay loop so that the

"Few chip" μP systems could also serve

Although a 1-chip device such as the 8748 offers the most compact approach to implementing a 1-shot lab application, there are many 2- and 3-chip alternatives that could make more sense for your requirements. Mostek's version of the 3870 equipped with a socket to carry a 2716 EPROM piggyback fashion comes quickly to mind; National plans a similar piggyback option for its 8048 family.

The ROMless version of the National Semiconductor COP 1-chip μC that we found so easy to use earlier in this series **(Ref 2)** would also prove quite suitable. And some of the standard masked-ROM parts can also be used in 2-chip systems; for example, you can pin-program the AMI S2000 family to ignore its internal resident ROM and fetch its instructions from an external EPROM.

Thanks to the availability of combination support chips (with assorted combinations of ROM, EPROM, RAM and I/O), many of the midrange μPs can also function as efficient 2- and 3-chip 1-shot systems. The 8085, F8, 8060, 6800 and 6500 families are some examples. We've also found that the Signetics 8X300 bipolar μP is ideal for direct TTL replacement. It has the speed and drive capability of TTL and requires only three chips for use in minimum systems.

Designers familiar with the 2900 bipolar-bit-slice-family parts ought to be able to concoct some useful small 1-shot systems with them. And the new 8-bit-wide super ECL from Fairchild and Motorola ought to answer the needs of designers working with megahertz signals.

For a more complete rundown on possible 1-, 2- and 3-chip hardware combinations, see **Ref 3,** which lists in its index the minimum number of chips needed in systems designed around the various μCs and μPs. Examination of this index reveals 15 1-chip possibilities, seven 2-chip combinations and five 3-chip systems.

As you start working with larger multichip systems, you might reach a point where you have enough memory—say 6k—to permit a switch to a higher level language. One of the very easy-to-use interactive, interpretive languages, such as BASIC or FORTH, would probably prove a logical choice for these 1-shot applications. Such a language permits you to use quickly written 100-statement programs, yet also tackle larger tasks.

At the recent National Computer Conference, for example, we examined XYBASIC, from the Mark Williams Co, Chicago. From president Robert Swartz's comments, this control-oriented BASIC for 8080 systems appears to have features that suit it for general laboratory use. For example, it has bit-manipulation capability and is intended for ROM-based end applications.

Before too long, at the rate 1-chip-μC memories are growing, you'll probably see 1-chip μCs with combined ROM and EPROM on chip. The ROM will contain resident high-level-language interpreters, and the EPROMs will be used for user application code—providing added flexibility for the designers of 1-chip-μC systems.

timer could be saved for more valuable use.

After each delay, the software checks to see whether the DUT has responded. In each case, the DUT output pulls the 8748's Test 1 input down, and an appropriate conditional-jump instruction that branches the program if the Test 1 input is true or false (JT1 and JNT1) is inserted after each delay. If the DUT does not respond to the 8748's ON command, program execution branches to a routine (DISPLY 1) that turns the yellow indicator LED ON steadily. If the DUT does not respond to the OFF command, program execution branches to another routine (DISPLY 2) that flashes the yellow indicator LED.

The tester records the number of successful DUT operations by inserting a call to a BCD-increment routine (BCD/INC) at a point in the cycling loop. This routine increments the results data string each time the DUT successfully goes through an ON/OFF cycle.

You'll need a minimum-hardware-cost readout

To observe the content of the results data string, we have thus far used the Prompt-48 monitor to examine the string's registers. But an actual stand-alone tester would be better served by some means of reading out the results from the tester itself. During tests, this feature could indicate whether the tester was functioning and show how many cycles the DUT had experienced. Such a resident readout would be absolutely necessary for checking components that might experience many millions of successful cycles over many weeks of *in-situ* testing.

The single-digit display denoted by dashed lines in **Fig 1** provides one such readout. It's simple, so it won't take up much design or construction time. And its single digit position permits the use of a large display to provide easy reading under field conditions.

Usually, this readout would show the results digit, which would be changing at about 1 Hz—just fast enough to show that the count was changing, but not so fast that an operator couldn't read its value. A common-anode LED display driven directly from one of the 8748 ports; it could possibly be equipped with series resistors to limit current drain.

The software to drive this display would include a lookup table to convert the decimal

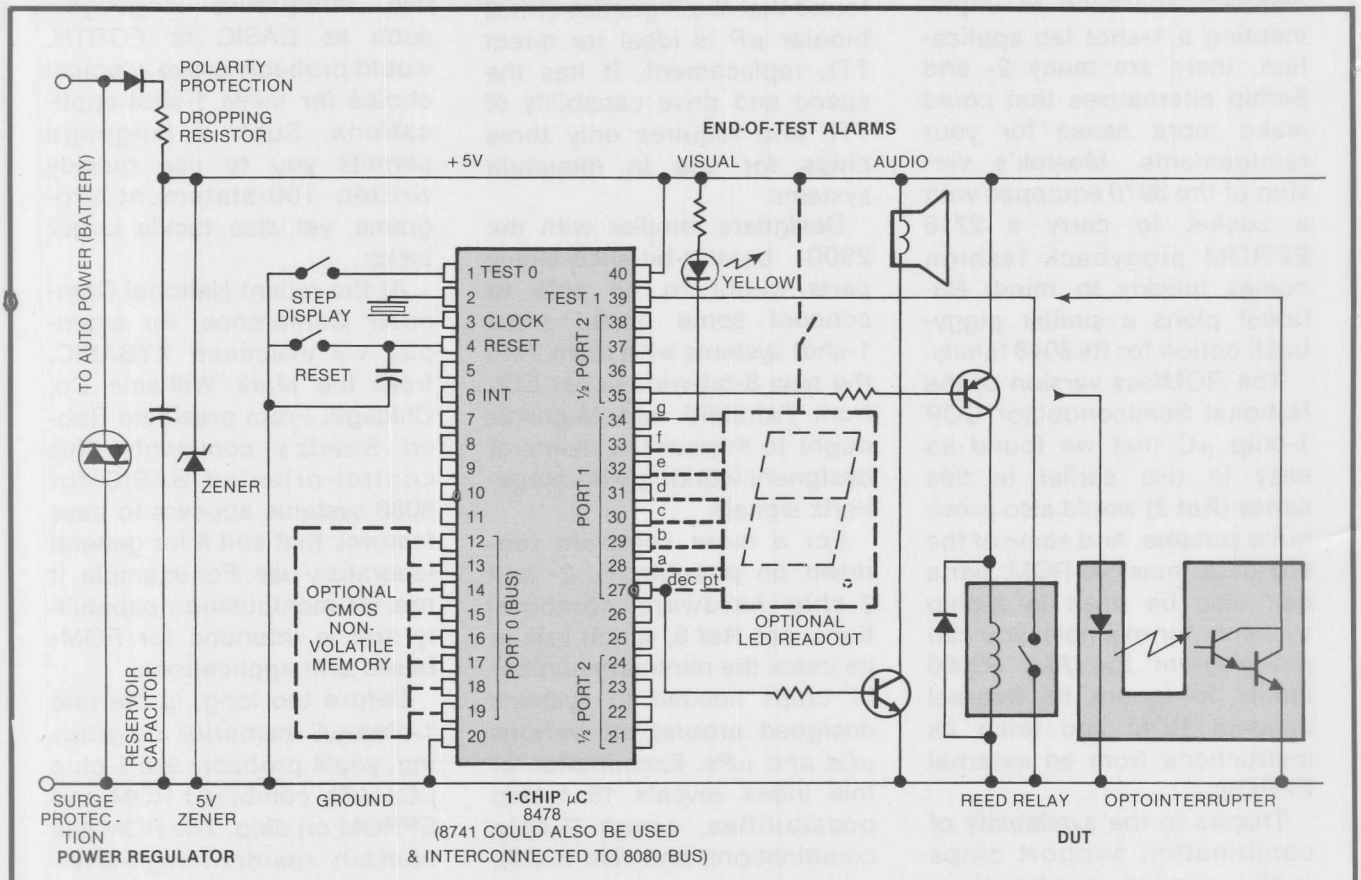

Fig 1—In a stand-alone test module using a 1-chip μC *for a 1-shot application, the 8748 is the sole logic element; the system requires none of the usual TTL formerly used for such a laboratory circuit. This example module assesses the reliability of critical or failure-prone parts such as semiconductor power devices, electromechanical relays or incandescent lamps under field conditions. The 8748 cycles the device under test (DUT) ON and OFF, checks that it has operated in each cycle and counts the number of successful operations; it flashes the display and sounds an alarm when the DUT fails. The power supply sketched in at the left is intended to operate from an automobile battery.*

Use of a 1-chip μC breeds familiarity for future projects

values in the results string to appropriate 7-segment patterns. It would also provide a means for stepping the display sequentially through all the digits of the results string, converting them from their packed-BCD form and looking up the 7-segment output patterns. In **Fig 1**, the push-button connected to the Test 0 input implements this display stepping.

The display's decimal point could be used to indicate the digit position currently displayed: For position 1, there could be a single flash of the decimal point followed by a 1-sec or greater separation delay, then another single flash, and so on. For position 2, there could be two flashes; for position 3, three flashes, etc.

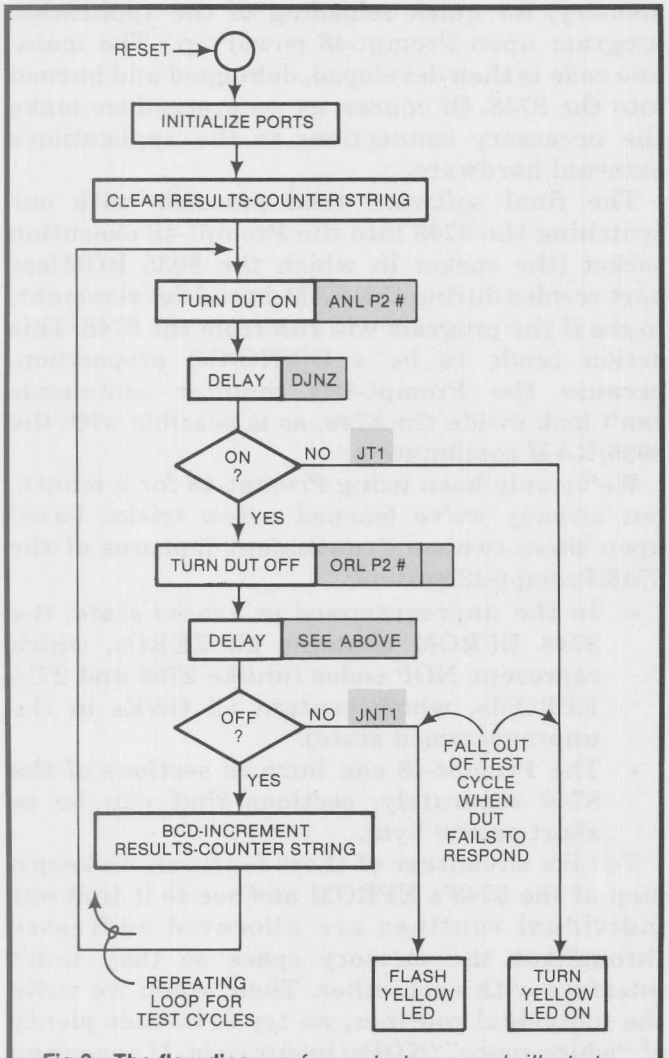

Fig 2—The flow diagram *for our tester example illustrates how simple and obvious the program for a 1-shot laboratory application can be. About 100 bytes of code and 1 week are required to implement this flow diagram; an experienced 8748 programmer could probably do it in half a day.*

A lab tool to build upon

As **Fig 1** shows, our example involves almost no hardware design at all. It could mount on a standard 100-mil-grid card, with the few required interconnections made by point-to-point wiring. Because of its simplicity, it should prove highly reliable.

Probably the most cost-effective method of field troubleshooting this tester would be to build several identical cards and just replace the whole module if problems developed. Actually, if component-life testing were your objective, running several tests simultaneously (to build up more statistical data and hedge against the possibility that a μC-system failure would abort a test run) would prove a better strategy. For this approach, duplicating the tester would be merely a matter of inserting more 8748s in the Prompt-48 (or other 8748 programmer) and taking a few extra minutes to copy the application program into them.

But the real advantage of using a 1-chip-μC approach would accrue to those labs regularly using it as a standard procedure over a period of years. The engineers and technicians in those labs would become so skilled at using the 1-chip μC that any of them could turn out programs like this one in less than a day. If—as is likely—each lab had a full-scale, minicomputer-like disc-based development system, the engineers and technicians would also be able to rapidly assemble much more sophisticated 1-shot programs while remaining within the time requirements management considers acceptable.

From the hardware standpoint, there should be much less wastage than with the TTL approach, for the 8748 can of course be erased and reused. The rest of the wiring is so simple that the card could probably also be reused. Perhaps this approach will hasten the disappearance of lab lockers jammed with ancient complex breadboards that nobody—not even their designers—now understands.

Breadboarding with Prompt-48 is easy

If a 1-chip μC is to replace TTL in 1-shot projects, an engineer must be able to program the device quickly. We found that we could use Intel's moderately priced ($2500) Prompt-48 development tool for the 8748 with only a few weeks of self training, and we were then able to work out and burn in small programs like our 100-line component-tester example within a week or less.

Intel downplays the Prompt-48 (**Fig 3**), urging users instead to turn initially to the $10,000-level full-blown Intellec Series of development tools. While the minicomputer-like Intellec systems do prove vastly superior for developing 1000-line programs, the simple, rugged Prompt-48 isn't

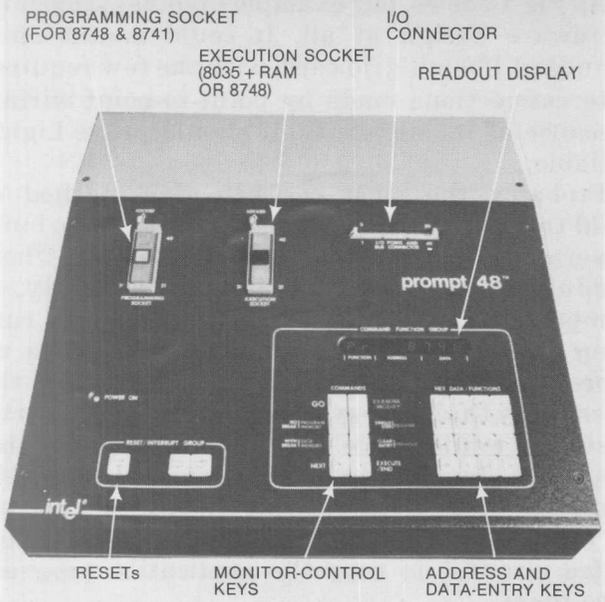

Fig 3—A development tool *for the 8748 1-chip μC, Intel's Prompt-48 proves well suited for working out short 100-byte programs and then burning the 8748's EPROM. Although relatively expensive compared with similar low-cost tools, it is well designed and rugged.*

that bad for the occasional user. We would advise a lab that wishes to explore the feasibility of using 8748s on a 1-shot basis to start with Prompt-48, for even if it did later purchase a larger Intellec station, it would still find the Prompt-48 useful.

We've found that we can work out programs on Prompt-48 routine by routine, burning each one into the 8748 as we go along. This procedure proves particularly helpful when developing 1-shot programs, because the Prompt-48 firmware doesn't include an assembler and you thus must resort to hand coding. We keep our routines short—no more than 10 to 20 lines each—and make generous use of subroutining to keep them isolated; sometimes 90% of an application resides in short subroutines, and the main line merely consists of calls to these routines. It's a sort of poor-man's structured programming.

After working up one of the routines using the coding sheet that Intel provides with the Prompt-48, we manually key the machine code into Prompt-48's internal RAM. The development system allows the use of the ROMless member of the 8048 family (the 8035) with RAM in this manner for initial code trials; the entry keys and associated monitor firmware are well designed and make manual entry virtually effortless.

We thus debug a short routine while it is in Prompt-48's RAM, using the monitor's single-step and break-point features. Then we transfer the routine from Prompt-48's internal RAM into the 8748's EPROM—an operation that takes a few keystrokes. At this point, the routine is safely burned into the 8748 chip.

In this manner, we can burn in all of a 1-shot application's subroutines into the 8748. The simple ones, like the time delay in the tester example, proceed quickly, taking no more than half a day. The more complex ones, like BCD incrementation, demand a bit more thought (and perhaps a search of the Intel application notes for a guide) but still require no more than several days.

After the subroutines are safely tucked into the 8748 EPROM, we turn our attention to the main-line code that unites them into a useful whole. But to work out and debug this code, we must be able to call the subroutines, so we key in the Prompt-48 command that copies the subroutines from the 8748 EPROM back into the Prompt-48 working RAM. (At this juncture, we are using the 8748's EPROM like a floppy-disc memory, for quick reloading of the application program upon Prompt-48 power-up.) The main-line code is then developed, debugged and burned into the 8748. Of course, as we proceed we make the necessary connections to the application's external hardware.

The final software trial proceeds with our switching the 8748 into the Prompt-48 execution socket (the socket in which the 8035 ROMless part resides during the RAM-based development) to see if the program will run from the 8748. This action tends to be a GO/NO-GO proposition, because the Prompt-48's monitor commands can't look inside the 8748, as is possible with the 8035/RAM combination.

We've only been using Prompt-48 for a month, but already we've learned a few tricks, based upon these two very convenient features of the 8748/Prompt-48 system:

- In the unprogrammed or erased state, the 8748 EPROM contains all ZEROs, which represent NOP codes (unlike 2708 and 2716 EPROMs, which contain all ONEs in the unprogrammed state).
- The Prompt-48 can burn in sections of the 8748 separately, sections that can be as short as one byte.

To take advantage of these features, we keep a map of the 8748's EPROM and see to it that our individual routines are allocated addresses throughout the memory space so they won't interfere with each other. Then, when we write the individual routines, we try to include plenty of "white space" (NOPs) in our code. If space and execution time are not critical—as they often are not in 1-shot applications—we might go to the extreme of putting a pair of NOPs between each pair of instructions. (We use NOPs in pairs because we then have room to later insert 2-byte JMP instructions. These JMPs occur to patches

that correct our inevitable coding errors, and they also prove handy for opening up the code to insert improvements.)

Thus, even though your code is frozen into the 8748's on-chip EPROM, you still can have the option of changing it in the future. Of course, you always have the option of dumping the 8748 code into the Prompt-48 working RAM, cleaning it up while you erase the 8748 and then reprogramming the 8748 EPROM with clean code. However, there is a psychological comfort in knowing that you can make small changes in the 8748 EPROM on a line-by-line basis.

But of course you must know from the start that the 1-chip μC you choose will handle your hardware and software requirements with room to spare. **Ref 1** relates the sad story of a user who, after delving deeply into an application, found the 8048 architecture too limited for his project. Keep in mind that the closer the magnitude of your application approaches the limits of a 1-chip μC's capabilities, the harder and longer you'll have to work to cram everything in.

One-shot feasibility depends on time

The case for using 1-chip μC's in 1-shot applications boils down to programming cost, which in turn depends on programming time.

Fig 4 shows a simplistic graphic model of the economics of using 1-chip μCs across the full spectrum of end-application volumes; it ranges from 1-shot applications at the left all the way out to the 10-million-unit volumes typical of consumer and automotive products. The extensive footnotes in **Fig 4** detail the underlying considerations.

Fig 4 graphs three main functions of application volume: One for engineering cost charged against each unit, one for the unit cost of masked-ROM 1-chip μCs and one for the unit cost of EPROM versions. This simple model illustrates the radically different economics of applying 1-chip μCs over today's wide range of end-product volumes. And it helps explain why 1-shot applications must be approached so differently from high-volume ones.

Taking this crude model at face value, you can see that several relationships exist between design cost and production volume. At very low volumes, where EPROM parts would be used, engineering cost dominates. But at about 1k, this cost falls below EPROM-μC cost. And at about 100k, engineering cost becomes negligible compared with the cost of the masked-ROM part used at that volume. (Beyond 100k, engineering cost fades into insignificance.)

As an aid in translating the engineering-cost

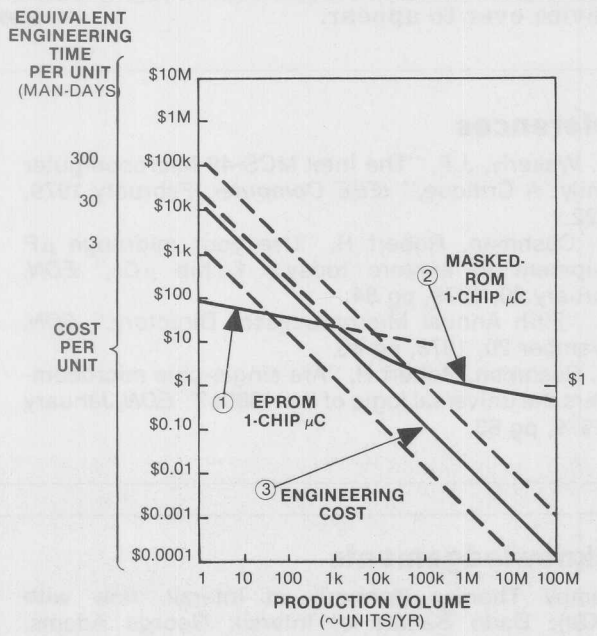

Fig 4—An economic model *for evaluating the cost-effectiveness of using an EPROM 1-chip μC spans a wide spectrum of production volumes, ranging from 1-shot applications at the left to multimillion-unit large-scale production at the right. At first glance, it would appear unthinkable to use any μC at low volumes because of high engineering costs. However, the cost of engineering comparable TTL solutions is also high. Thus, the price differential between an EPROM 1-chip μC (which might cost nearly $100 in single quantity) and a handful of TTL parts (which might cost a few dollars) becomes an inconsequential consideration compared with the engineering-manpower costs of either approach.*

NOTES

1 — EPROM 1-chip-μC curve starts (to left) at the current single-quantity price for the 8748, which is $88. It slopes downward gradually toward a price of $10 per device at 100k volume, based on informal comments from EPROM 8748 suppliers on expected volume pricing in the 1980s. The curve becomes dashed at about 5k volume because 5k is as high as current EPROM order volumes have gone, according to Intel. However, it continues out to higher volumes because several future suppliers of EPROM 1-chip devices (TI and Motorola) believe there will be a higher volume market for EPROM 1-chip μCs, arising when users in the unpredictable automotive and appliance fields will want to stock standard unprogrammed EPROM 1-chip devices and not commit them to application programs until the last minute.

2 — Masked-ROM 1-chip-μC curve starts out at the left as a dashed line and follows roughly the same reciprocal cost-per-volume relationship as the engineering-cost curve. But when it reaches about 10k volume, this curve flattens out. Its bent shape is explained by the fact that at lower volumes, mask charges for the part dominate (and the curve is dashed because few customers order masked-ROM parts at less than about 5k volume), while at higher volumes, materials and processing costs dominate. This curve goes below $1 at more than 1 million volume to account for the likely possibility that very high-volume users might resort to unconventional packaging (bare chips on pc boards) and complete automation.

3 — Engineering costs are summed up in a straightforward reciprocal curve, one that's a solid line throughout its length. We assume that as much engineering effort is expended for one copy as for 10 million. The magnitude of the engineering cost is computed on the simple-minded assumption that each line of code requires $10 worth of engineering effort; because the basic 1-chip μC has a 1k ROM, this amounts to $10k cost per unit. This curve is widened out by an order of magnitude in each direction to account for the true situation. With an engineering man-hour costing a company $30 to $50 per hour (the engineer receives half), it is unlikely that programming — which is said to proceed at about one line of code per hour — can be achieved for just $10 per line. On the other hand, the very short, simple programs adequate for some of the 1-shot applications (such as this article's example) can be written at the rate of 10 lines/hr.

Reprinted from **EDN** August 5, 1979

EPROM devices won't always be hard to get

Currently, all modern 5V EPROM devices are not only expensive, but also in very short supply. This fact holds true not only for 1-chip devices with on-board EPROMs (like the 8748), but also for EPROM chips (like the 2716) that would be used in multichip system designs.

The problem? EPROMs are large-area devices that are difficult to fabricate; only a few suppliers, notably Intel, have been able to manufacture them in quantity. Meanwhile, as an ever-widening group of users has sprung up, the demand for EPROM-based devices has risen exponentially: Everyone needs them in development labs and in short-run production, and some users also find that they want to continue to use EPROMs as they achieve volume production.

We have conducted a spot check of distributors and suppliers to see when they think this situation might ease. Intel sources say that increased yields and production capacity should ease the supply picture by the fourth quarter of this year and that stock should begin to build up on distributor shelves during early 1980. And some of the second sources for the 8748 and similar 1-chip EPROM-programmed parts are now successfully sampling devices and expect to achieve volume production by 1980.

Distributors are more pessimistic; they appear cynical about supplier promises of availability. Some refuse to quote price and availability at all on EPROM parts "until the situation straightens out."

Finally, a spokesman for Data I/O, the leading PROM- and EPROM-programmer manufacturer, says that the firm's universal programmers, such as System 19, have been selling well because customers want to be able to switch to alternative types of PROMs (even bipolar units) when they can't obtain the EPROM devices they would prefer.

However, all sources feel that because of high device prices, continually mounting demand and many new sources in the US and Japan, the supply situation should be resolved some time in 1980.

figures into more meaningful time figures, we've shown the rough design-time equivalent alongside some of the unit costs on the vertical axis of Fig 4. We've also widened out the engineering-design-cost curve by an order of magnitude in each direction to account for the wide disparity of design effort applicable when ranging from quick-and-dirty 1-shot laboratory applications to highly competitive mass-produced end products.

While the nominal time that might be invested in a 1-chip-μC application design might be on the order of a month or so, as much as 1 man-yr might be warranted for a very high-volume product, with several engineers being assigned to keep the actual elapsed time within reason. The additional effort would be directed at honing down the hardware production costs and increasing the performance crammed into the software.

Goodby to 1-chip μCs for now...

With this article, we come to the end of our 1979 look at 1-chip μCs. We've come full circle, for we started out by considering these devices as possible TTL replacements (**Ref 4**), then examined their use in high-volume applications and now have examined their use in 1-shot designs, which brings us back to TTL replacement.

The 1-chip-μC story is hardly over, though, and you can be sure we'll be returning to it. These devices now appear to be the unruly mob of the μP revolution; they are finding use in more places and in larger numbers more quickly than anyone had expected. Several suppliers have told us that their capacities have been signed up for the next 6 months. These devices might in fact prove the most successful class of OEM electronic device ever to appear. **EDN**

References

1. Wakerly, J F, "The Intel MCS-48 Microcomputer Family: A Critique," *IEEE Computer,* February 1979, pg 22.
2. Cushman, Robert H, "Use your midrange μP equipment to explore today's 1-chip μCs," *EDN,* February 20, 1979, pg 84.
3. "Fifth Annual Microprocessor Directory," *EDN,* November 20, 1978, pg 86.
4. Cushman, Robert H, "Are single-chip microcomputers the universal logic of the 1980s?" *EDN,* January 5, 1979, pg 83.

Acknowledgements

Thampy Thomas (formerly of Intersil, now with ELXSI); Dado Banato of Intersil; George Adams, Steven Weisman, John Wharton, Lionel Smith, Don Phillips and Larry Jordan, all of Intel; and Tom Harper of National Semiconductor provided invaluable assistance in the preparation of this article.

intel®

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 (408) 987-8080

Printed in U.S.A./TP-9/0979/30K